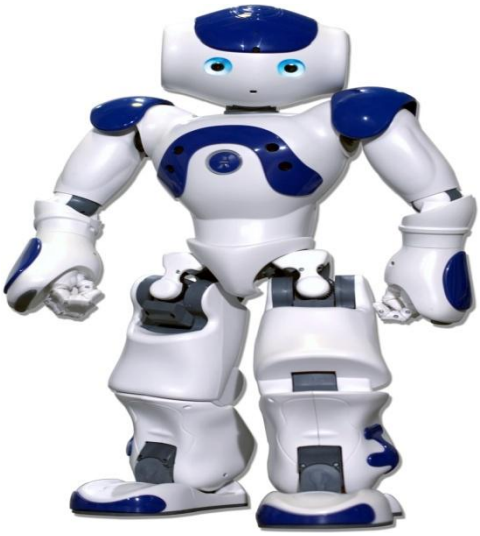
Plateforme Robots



Réalisé par l'équipe N° 2 :

BELLINI Quentin
GNANAKULENTHIRAN Anitha
GOVINDEN Johana
MEZINE Ahcene
TIMZOUERT Chabane

Equipe de suivi : M. Thierry HAMON & Mme Sophie TOULOUSE.

Client : Mme TOULOUSE

INFO2 2011/2012

31/05/2012

Remerciements :

Nous tenons à remercier particulièrement **Mme Sophie TOULOUSE** et **M. Thierry HAMON** pour nous avoir encadrés et aidés tout au long de ce projet.

Nous remercions **M. Jean-Pierre BELLIN** et l'équipe du **CRIG** de **Sup'Galilée** d'être disponibles et d'avoir répondus à nous diverses questions techniques.

Merci à **M. Rami TEMIN** de l'université Paris Descartes pour ses conseils au début de ce projet.

Nous remercions toute personne ayant participée ou contribuée de prêt ou de loin à la réussite de ce projet.

Historique des modifications/mises à jour du document :

Date	Modifications	Page(s)	Auteur(s)
30/11/2011	Première version : réponse définitive		Equipe
12/12/2011	-Page de garde -Ajout de l'historique -Diagramme de Gantt	1,3, 9	Ahcene
26/12/2011	Description et fonctionnement du robot Lego Mindstorm	10,11,12	Johana
16/01/2012	Descriptif des solutions existantes	13	Equipe
20/01/2012	Mise à jour		Equipe
29/01/2012	Manuel d'installation NXC	Manuel_NXC	Quentin
31/01/2012	**Livraison de la solution NXC**		
10/02/2012	Solution NXC	13,14,15	Quentin,Chabane
12/02/2012	Manuel d'installation leJOS	Manuel_leJOS	Ahcene
12/02/2012	**Livraison de la solution leJOS**		
15/02/2012	Annexe : vocabulaire	31	Johana, Anitha
23/02/2012	Solution **nexttool**	16	Equipe
10/03/2012	Mise à jour diagramme de Gantt	9	Ahcene
24/03/2012	-Mise à jour du manuel d'installation nxc -Guide de programmation NXC	Manuel_NXC Guide_NXC	Equipe
20/04/2012	-Mise à jour du manuel nxc (Liaison USB) -Mise à jour Guide_NXC (ajout exemples)	Manuel_NXC Guide_NXC	Equipe
10/05/2012	Tests sur des **machines virtuelles** (squeeze, wheezy, ubuntu) avec **VirtualBox**	Solution_NXC	Equipe
22/05/2012	Test de la solution Bluetooth Test de l'exemple de jeu	Bluetooth Exemple nxc	Equipe
27/05/2012	Mise à jour du rapport	Rapport	Equipe
28/05/2012	**Livraison de la version finale de la solution NXC :** -Manuel d'installation NXC -Package NXC avec vidéos tests -Guide de programmation NXC -Exemple de jeu test		Equipe
29/05/2012	**Soutenance et présentation**		Equipe

Table des matières

I. Introduction …… 5

II. Présentation du projet …… 6

Objectif …… 7

III. Mise en œuvre …… 7

1) Technique …… *7*

2) Matérielle …… *7*

IV. Mise en œuvre humaine …… 8

V. Planification et découpage de tâches …… 9

VI. Description et fonctionnement du robot Lego…… 10

VII. Descriptif des solutions existantes …… 13

VIII. Solution NXC …… 14

- Problème de droit et de liaison USB …… 16
- Test de la solution bluetooth …… 16

IX. Test sur des machines virtuelles avec VirtualBox (version 4.0.2) …… 19

X. Solution pour la programmation en Java …… 20

XI. Conclusion …… 29

XII. Annexes : Vocabulaire, liens et sources, CV …… 30

I. Introduction :

Ce projet est le premier de son genre qu'on propose aux élèves ingénieurs de spécialité informatique à **Sup'Galilée**, ce qui représente pour nous un vrai défi technique et un véritable challenge pédagogique.

En plus de son aspect ludique qui permettrait aux futurs et chanceux étudiants qui travailleraient avec ces robots d'apprendre tout en s'amusant, c'est une occasion pour notre équipe d'explorer un autre univers, ou l'application de nos connaissances informatiques s'avère très intéressante.

C'est un projet qui nous motive, qui répond à notre curiosité scientifique. Sa finalité est de donner une nouvelle approche à l'apprentissage de l'informatique au sein de l'école.

Voir un robot bouger après lui avoir installé un programme informatique, c'est comme donner une âme à cette machine et lui permettre de vivre.

II. Présentation du projet :

1) Objectif :

L'objectif de l'équipe est d'offrir des plateformes de travail documentées et facilement ré-installable sur plusieurs environnements **(Linux)**, afin de permettre aux étudiants de découvrir la programmation et les principes de l'informatique par l'utilisation de des robots.

Notre mission consistera donc à s'approprier les technologies disponibles et à les faire évoluer pour les rendre les plus claires et complètes possibles, dans l'optique d'une utilisation pédagogique en privilégiant les solutions **open source**.

Idéalement nous pensons pouvoir offrir deux environnements pour chaque robot l'un orienté programmation impérative en **C** et l'autre programmation orientée objet en **Java** et éventuellement **C++.**

En attendant de recevoir les autres robots, on se focalisera sur le robot « **Mindstorm** » et on commencera par mettre en place une plateforme permettant la programmation par le langage C, afin de se familiariser avec le Robot et d'avoir une idée précise des problèmes que l'on pourrait rencontrer.

D'ailleurs dans cette perspective, nous nous somme rapprochés des étudiants en Télécom qui ont l'habitude de travailler et de programmer des robots, ainsi que de leur technicien **M. BELIN** et de l'association **CRIG** de **Sup'Galilée** qui nous ont renseigné d'avantage et s'impatientent de collaborer avec notre équipe pour nous assister lors des différents tests pratiques.

Une fois les tests validés, on s'intéressera à la deuxième plateforme qui permettra la programmation en java et éventuellement en C++.

Après réalisations des différents tests, nous fournirons le package d'installation et la documentation correspondante à l'environnement en question, ce qui aboutira à la livraison, à chaque étape, d'une brique pour tester les fonctionnalités développées.

Et dès réception des autres robots, on testerait les environnements sur chaque plateforme.

III. Mise en œuvre :

1) Mise en œuvre technique :

Tout d'abord nous effectueront des recherches détaillées sur les différentes solutions disponibles pour le robot «**LEGO Mindstorm** » que nous testerons avec les interfaces : **Labview** et **RobotC.**

Afin de distinguer les environnements qui répondent mieux à nos besoins, nous effectuerons une revue des qualités et des défauts de chaque solution pour pouvoir les comparer.

Ensuite nous choisirons les deux environnements les mieux adaptés et nous vous informerons de nos choix, tout en explicitant les raisons de ces choix.

Nous nous sommes déjà renseignés sur certains environnements comme **RobotC** et **NXTGCC** pour la programmation **C, ntxOSEK** pour le **C++** et **leJOS** pour **Java**. Nous connaissons par exemple la procédure à suivre pour utiliser l'environnement **leJOS** en passant notamment par l'ide **Eclipse** que plusieurs membres de notre groupe maîtrisent déjà.

Une fois les environnements choisis, nous chercherons à les packager de manière à rendre leur installation la plus simple possible (avec la création de script ou de makefile par exemple), en offrant éventuellement une interface graphique dédiée a cette installation.

De plus, nous avons déjà dû mettre en place des interfaces homme machine pour différents projets antérieurs, ce qui nous à permis de maîtriser les bibliothèques **Qt** et **GTK** afin d'obtenir des interfaces claires et simples d'utilisation.

Comme nous l'avons déjà explicité, nous pourrons de plus, profiter de l'expérience des membres de l'association **CRIG** pour tester ces robots de manière efficace.

2) Mise en œuvre matérielle :

Un robot **Mindstorm** est mis à notre disposition en début de ce projet, et d'autres (**Nao** et **Vex** ou **Bioloide**) nous seront éventuellement fournis pour pouvoir tester nos solutions sur différentes plateformes.

Une licence **RobotC** monoposte nous a été fournie, et le CD d'installation de l'interface **LabView** pour pouvoir tester le Robot **Mindstorm**.

Chaque membre de l'équipe dispose d'un PC portable déjà prêté par l'école pour la durée de la scolarité.

De plus, nous aurons accès aux salles et matériels déjà utilisés par les élèves ingénieurs Télécom et par l'association **CRIG** pour faire les tests de ramassage de balles par exemple.

IV. Mise en œuvre humaine :

Notre équipe est constituée de cinq élèves ingénieurs en deuxième année d'informatique. Nous pouvons vous offrir **25 j/h** de travail en moyenne, par personne, soit un total de **125 j/h** en moyenne pour l'ensemble du projet (hors réunions et suivis).

Les membres de l'équipe sont les suivants :

BELLINI Quentin
GNANAKULENTHIRAN Anitha
GOVINDEN Johana
MEZINE Ahcene
TIMZOUERT Chabane

Vous trouverez en annexe le curriculum vitae de chaque membre du groupe.

Pour l'organisation du travail, la plupart des membres de l'équipe ont déjà travaillé ensembles sur différents projets, ce qui facilite la répartition des tâches en fonction des compétences de chacun.

Pour notre projet, nous avons choisi une méthodologie agile qui est **SCRUM**, avec

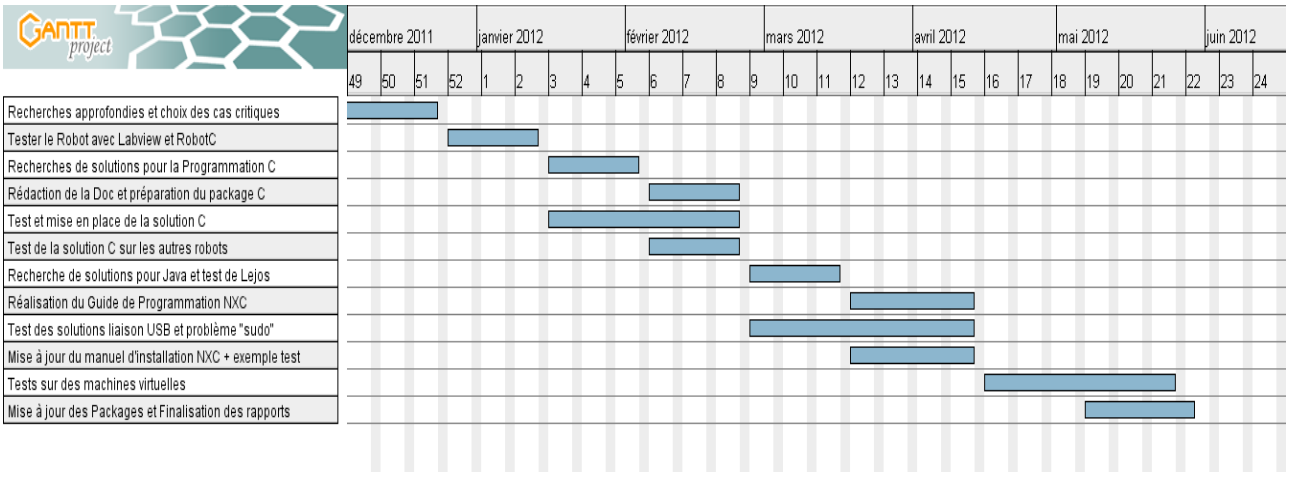
M. MEZINE Ahcene comme **Scrum Master** pour veiller à ce que les valeurs de **scrum** seront bien respectées.

Nous travaillerons avec des **Sprints** (itérations) de **trois semaines**, ce qui nous permettra de vous livrer une brique à chaque itération.

V. Planification et découpage de tâches :

Nous vous présentons une première planification provisionnelle pour le déroulement de notre projet avec un découpage des tâches qui susceptible d'aménagements et de modifications vu que l'avancement de notre projet dépend des résultats des différentes recherches et de la mise en place des solutions trouvées.

Néanmoins, il nous semble réaliste et cohérent. Le choix de travailler avec la méthodologie **SCRUM** nous permet une grande souplesse pour adapter notre planification en fonction de vos besoins et des éventuelles difficultés qui peuvent être rencontrées.


GANTT project
décembre 2011
janvier 2012
février 2012
mars 2012
avril 2012
mai 2012
juin 2012
49 50 51 52 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Recherches approfondies et choix des cas critiques
Tester le Robot avec Labview et RobotC
Recherches de solutions pour la Programmation C
Rédaction de la Doc et préparation du package C
Test et mise en place de la solution C
Test de la solution C sur les autres robots
Recherche de solutions pour Java et test de Lejos
Réalisation du Guide de Programmation NXC
Test des solutions liaison USB et problème "sudo"
Mise à jour du manuel d'installation NXC + exemple test
Tests sur des machines virtuelles
Mise à jour des Packages et Finalisation des rapports

Ce diagramme représente nos souhaits de réalisation avec une vision très optimiste sur l'avancement du projet. Néanmoins, certains objectifs dépendent des résultats des recherches et des solutions trouvées, ainsi que des moyens mis à notre disposition (autres robots).

VI. Description et fonctionnement du robot Lego Mindstorm :

Lego Mindstorms NXT est un jeu de construction et de robotique présenté par Lego en 2006. Il succède à la gamme Lego Mindstorms RCX.

1. Description :

Cette partie décrit le robot et ses accessoires le permettant de se déplacer, de capter les informations autour de lui et de réagir en conséquence, mais également la communication avec un ordinateur.

1.1 Le cerveau :

Le robot est composé d'une **brique NXT** qui est une brique contrôlée par un ordinateur et qui est le cerveau du robot mindstorm.

Voici un exemple de brique ci-dessous :

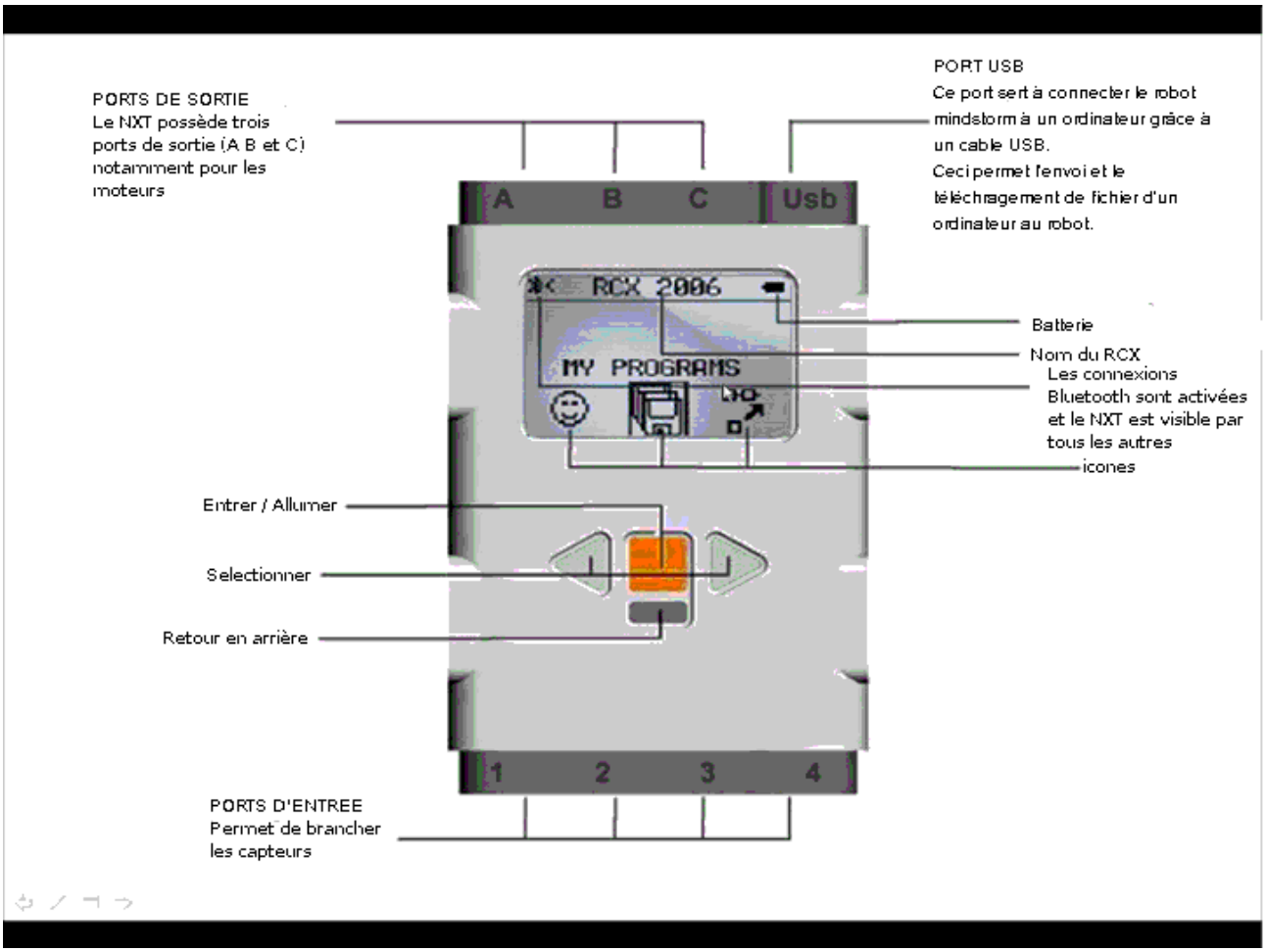
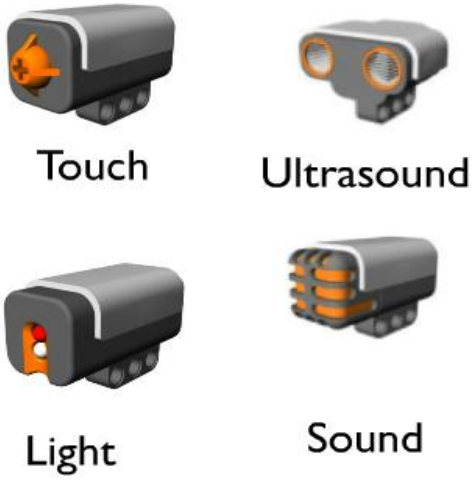


fig 1. Brique NXT

1.2 Les capteurs :

Le robot est composé de plusieurs capteurs :



- Les capteurs tactiles : ces capteurs aux nombres de 2 permettent au robot de réagir face à des obstacles
- Un capteur sonore : ce capteur permet de réagir au niveau sonore
- Un capteur photosensible : ce capteur permet de réagir aux variations de couleurs et de niveau de lumière
- Un capteur d'ultrasons : ce capteur permet au robot de mesurer la distance entre lui et un objet et de réagir aux mouvements

fig.2 Capteurs

1.3 Servomoteur :

C’est un moteur qui permet au robot d’effectuer des mouvements et de se déplacer.

Plus précisément, les servomoteurs, ici, représentent les bras et les jambes du robot.

Voici le servomoteur :

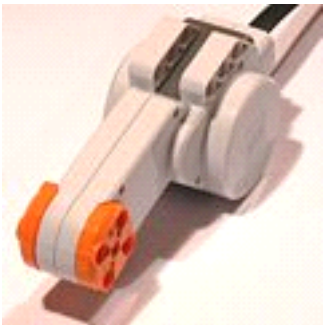


fig 3. Un servomoteur

2. Le fonctionnement :

Le fonctionnement du robot se fait ainsi :

Des tests peuvent être effectués directement depuis la brique NXT, sans avoir besoin d'ordinateur. Le robot peut, par exemple, reculer, avancer, mais il faut s'assurer que les capteurs et les moteurs (bras et jambes) sont branchés aux bon ports.

Les fichiers déjà compilés/chargés et disponibles sur la brique NXT peuvent être directement exécutés.

Des programmes peuvent être crées grâce à un environnement de travail ou d'un IDE. L'environnement choisi pour l'instant pour tester **est LabView**(logiciel de développement basé sur un langage de programmation graphique appelé **langage G**.)

Dans la fig1. , la brique est dotée d'un port USB. Le transfert et le téléchargement de programmes crées grâce à l'environnement de travail se font grâce à une connexion avec un câble USB. Les transferts de programmes peuvent également s'effectuer à travers une connexion bluetooth.

VII. Descriptif des solutions existantes :

En voici un tableau récapitulatif :

Caractéristiques	Linux	Windows	Mac OS	Langage	Firmware	IDE	Bluetooth
NXT-G Retail	Non	Oui	Oui	Graphique	Stadard	Oui	Oui
NXT-G edu	Non	Oui	Oui	Graphique	Stadard	Oui	Oui
RoboLab 2.9	Non	Oui	Oui	Graphique	Stadard	Oui	Non
NBC	Oui	Oui	Oui	Assembleur	Stadard	Oui	Oui
NXC	**Oui**	**Oui**	**Oui**	**Not Exactely C**	**Stadard**	**Oui**	**Oui**
RobotC	Non	Oui	Non	C	Stadard	Oui	Oui
LabView	Non	Oui	oui	Graphique	Stadard	Non	Oui
leJOS NXJ	oui	Oui	oui	Java	Stadard	Oui (eclipse)	Oui
leJOS OSEK	Non	Oui	non	AINSI C	Stadard	Oui (eclipse)	Oui
pbLua	oui	Oui	oui	Lua	Stadard	Non	Oui

Source : http://www.teamhassenplug.org/NXT/NXTSoftware.html

VIII. Solution pour le langage C :

❖ NXC

Pour utiliser du C sur le robot Mindstorm nous nous sommes tournés vers le **NXC**. Ce langage proche du C a été développé pour la brique **NXT** et est donc parfaitement adapté à nos besoins. De plus on trouve une communauté active autour de ce langage et il possède une documentation très complète. Par ailleurs, il nous a été recommandé par un passionné de robot Mindstorm.

Nous avons alors cherché les solutions permettant d'utiliser ce langage.

➢ Windows :

Sous Windows, il suffit d'installer **Bricx Command Center**, un IDE dédié au langage NXC (mais aussi à d'autres comme leJOS par exemple) et permet de compiler et d'envoyer des programmes NXC sur le robot. De plus, il permet aussi de changer le firmware de la brique.

➢ Linux :

Sous linux il existe plusieurs solutions. Néanmoins toutes ces solutions nécessitent un logiciel en commun. En effet il est nécessaire de compiler le code **NXC** en code **RXE** avec le compilateur **NBC** (présent dans le package). Ce compilateur est très simple à installer et marche parfaitement (comme le compilateur gcc, il signale les erreurs lors de la compilation).

Ensuite, pour transférer le code compilé sur le robot nous avons testé plusieurs solutions :

▪ Première solution : t2n

Tout d'abord nous nous sommes tournés vers t2n (talk to NXT) pour sa facilité d'installation. Grâce à un tutoriel clair nous avons réussi à envoyer notre code sur le robot. Néanmoins nous avons du faire face à plusieurs difficultés. Tout d'abord il nous a fallu passer root pour réussir à envoyer le fichier sur le robot (c'était un cas de figure prévu par le tutoriel). Mais nous avons aussi rencontré une erreur qui n'était pas prévu par le tutoriel :

```
Error :
usbnxt: upload: can'y initiate upload
(reply=0x02,status=0xffffff92,error=0x0000)
```

Après des recherches sur internet nous avons vu que cette erreur provenait d'une incompatibilité entre t2n et les firmware récent de la brique NXT (à partir du **firmware 1.05** pour être précis). En installant la version 1.05 du firmware officiel lego nous avons donc réussi à faire fonctionner cette solution.

Néanmoins, trouvant cette solution fastidieuse nous avons cherché s'il n'existait pas d'autres solutions fonctionnant sur toutes les versions du firmware.

Nous avons alors trouvé les solutions **py-t2n** et **NeXT Tools** (et NeXT Command Center). Ces solutions nous ont aussi demandées de passer en root pour fonctionner, néanmoins ce souci vient d'un problème de configuration de nos sessions en tant qu'administrateur et pas des solutions proprement dites.

▪ Deuxième solution : py-t2n

La solution **py-t2n** est très semblable à t2n (elle est écrite en python alors que t2n est écrite en C++). Cette solution est fonctionnelle, sous une ancienne version **Ubuntu**.

Néanmoins, nous avons rencontrées des erreurs lors de l'installation du package python sous **Debian**.

```
Traceback (most recent call last):
File "/usr/local/bin/py-t2n", line 22, in <module>
import nxt.locator
ImportError: No module named nxt.locator
ImportError: No module named nxt.brick
```

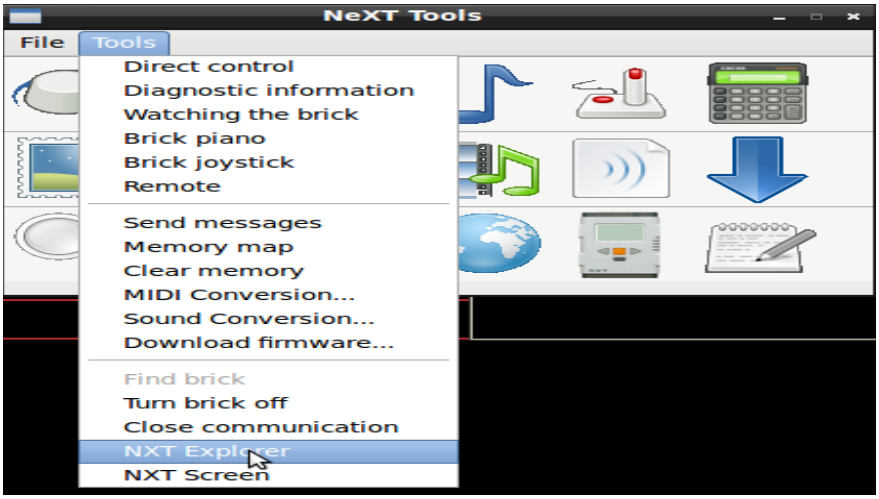
En absence de solution de résoudre ce problème, nous avons cherché et testé d'autres solutions.

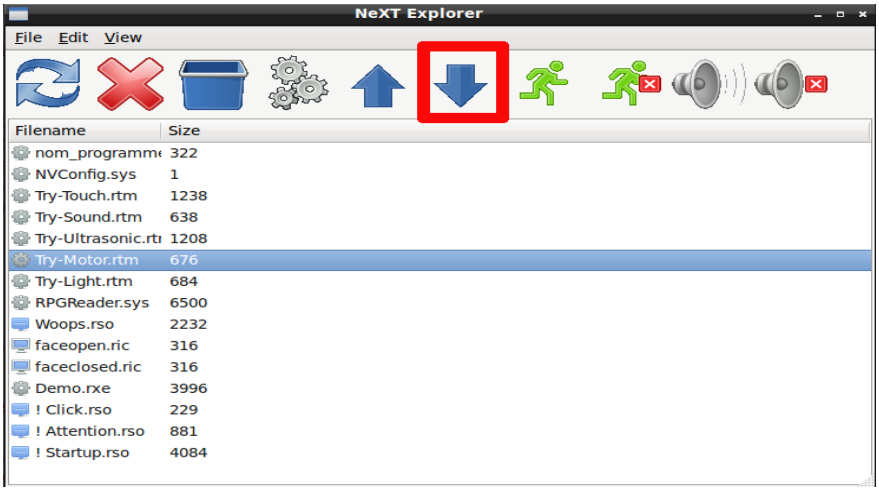
- **Troisième solution : NeXT**

La solution NeXT Tools (et NeXT Command Center) est une interface graphique reprenant les fonctionnalités de Bricx Command Center sous linux. Cette solution est donc très complète néanmoins elle entraine beaucoup d'erreurs, ainsi elle propose une fonctionnalité permettant de changer le firmware du robot mais l'utilisation de cette fonction entraine un plantage sévère du robot (il faut forcément retirer les piles du robot pour le refaire fonctionner). Il est aussi possible d'utiliser un éditeur de texte intégré à NeXT Command Center offrant des fonctions de compilation et d'envoi sur le robot.

L'envoi du programme fonctionne mais entraine une erreur lors de l'exécution (lorsque que l'on tente de lancer ce programme sur le robot on obtient un message d'erreur). Les fonctionnalités de contrôle direct du robot fonctionnent correctement ainsi que l'explorateur de fichier sur le robot.

C'est cet explorateur qui va nous permettre de charger correctement notre programme sur le robot en suivant le processus suivant :





On choisit le fichier compilé (d'extension .**rxe**), grâce à la fenêtre d'exploration, qu'on envoie directement sur le robot. Cette solution fonctionne et est simple à installer pourtant son utilisation demeure un peu laborieuse puisqu'elle demande de compiler le fichier en ligne de commande pour ensuite utiliser une interface graphique.

- **Quatrième solution : nexttool (en ligne de commande)**

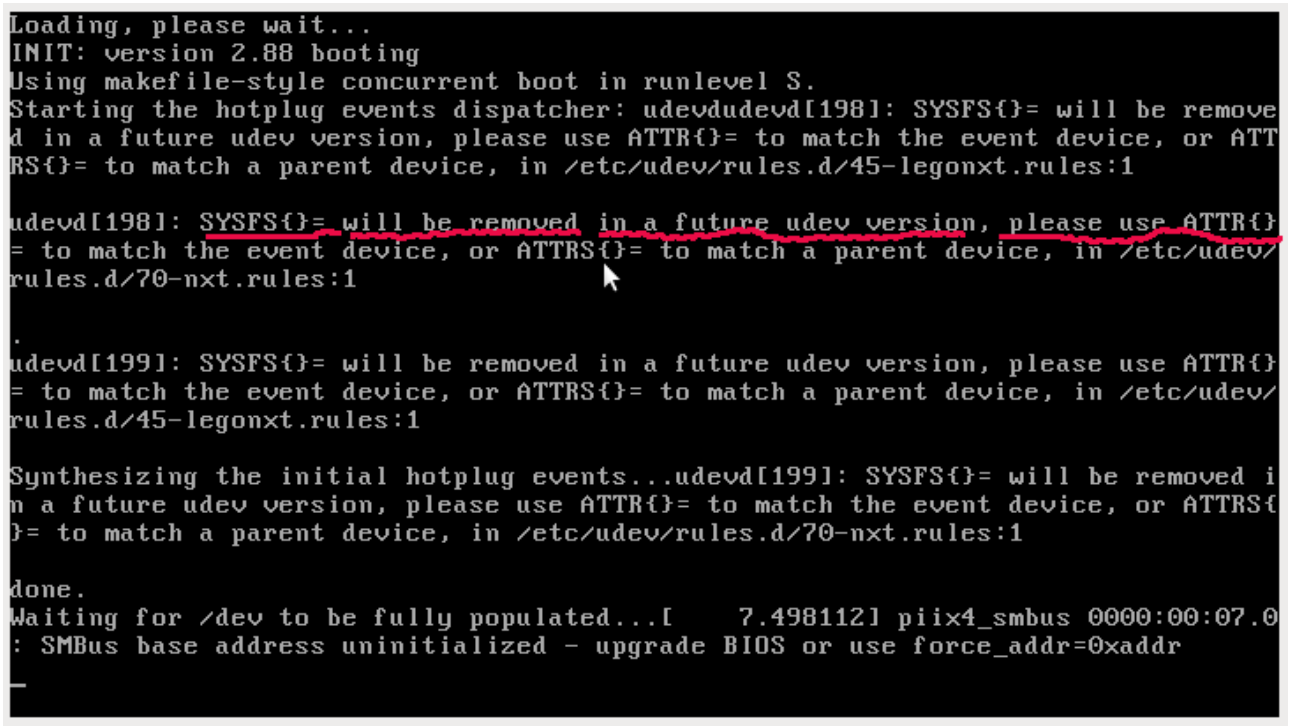
C'est la version en ligne de commande **NeXT**, il s'agit exactement du même programme développé par **J.Hansen**. On a opté pour cette solution car elle est facile à utiliser, et permet d'envoyer des fichiers compilé **.rxe** vers la brique **NXT**. Par contre, on retrouve les mêmes bugs que la version graphique, en ce qui concerne le changement de firmware.

D'autre part, on s'est rendu compte que le compilateur **nbc** permet aussi de faire le transfert de fichiers vers la brique. (voir manuel, pour plus de détails)

❖ Problème de droit et de liaison USB :

Les solutions décrites précédemment ne peuvent être utilisées qu'on mode **super-utilisateur (root),** et la solution trouvée est de créer un groupe des utilisateurs du robot à qui on donne le droit d'utilisation de la liaison **USB** pour communiquer avec le robot ***(voir manuel d'installation de la solution NXC, partie 4, page 9)***.

Pendant les tests de cette solution, on a constaté que la configuration du fichier **.rules** peut être différentes d'une version Linux à une autre (ancienne/nouvelle) d'où l'intérêt de bien observé les messages d'erreur lors du redémarrage de la machine.


Loading, please wait...
INIT: version 2.88 booting
Using makefile-style concurrent boot in runlevel S.
Starting the hotplug events dispatcher: udevdudevd[198]: SYSFS{}= will be remove
d in a future udev version, please use ATTR{}= to match the event device, or ATT
RS{}= to match a parent device, in /etc/udev/rules.d/45-legonxt.rules:1
udevd[198]: SYSFS{}= will be removed in a future udev version, please use ATTR{}
= to match the event device, or ATTRS{}= to match a parent device, in /etc/udev/
rules.d/70-nxt.rules:1
udevd[199]: SYSFS{}= will be removed in a future udev version, please use ATTR{}
= to match the event device, or ATTRS{}= to match a parent device, in /etc/udev/
rules.d/45-legonxt.rules:1
Synthesizing the initial hotplug events...udevd[199]: SYSFS{}= will be removed i
n a future udev version, please use ATTR{}= to match the event device, or ATTRS{
}= to match a parent device, in /etc/udev/rules.d/70-nxt.rules:1
done.
Waiting for /dev to be fully populated...[7.498112] piix4_smbus 0000:00:07.0
: SMBus base address uninitialized - upgrade BIOS or use force_addr=0xaddr

❖ Test de la solution bluetooth :

En fin de ce projet, nous avons voulu utiliser la liaison **bluetooth** pour communiquer avec le robot, et en voici les détails de la solution testée :

✓ Configuration de *rfcomm* :

Installer le package ***bluez-utils*** **(ou *bluetooth*)** et activer le Bluetooth sur la brique NXT.

Récupérer l'adresse Bluetooth du NXT avec la commande ***hcitool*** :

```
hcitool scan
Scanning ...
    00:16:53:05:80:2D NXT
```

Ajouter dans le fichier ***/etc/bluetooth/rfcomm.conf*** (en **root**) :

```
rfcomm0 {
    # Automatically bind the device at startup
    bind yes;
    # Bluetooth address of the device
    device 00:16:53:05:80:2D;
    # RFCOMM channel for the connection
    channel 1;
    # Description of the connection
    comment "NXT";
}
```

Pour configurer plusieurs NXT, il suffit de créer autant d'entrées que de NXT en incrémentant la valeur de l'indice ***rfcomm*** (par exemple, rfcomm1, rfcomm2, etc.)

Editer un fichier nomé ***41-legonxt.rules*** :

```
SUBSYSTEM=="usb", ACTION=="add", ATTR{idVendor}=="0694",
ATTR{idProduct}=="0002", SYMLINK+="legonxt-%k",
KERNEL=="rfcomm[0-9]*", GROUP="legonxt", MODE="0660",
RUN+="/etc/udev/legonxt.sh"
```

Copier le fichier ***41-legonxt.rules*** dans dans le répertoire ***/etc/udev/rules.d/*** (en mode root):

```
cp 45-legonxt.rules  /etc/udev/rules.d/
```

Redémarrer le service Bluetooth (en **root**) :

```
/etc/init.d/bluetooth restart
```

Vérifier que rfcomm fonctionne avec la commande ***rfcomm***:

```
rfcomm
  rfcomm0: 00:16:53:05:80:2D channel 1 clean
```

Vérifier que le device configuré est présent et que les permissions sont adéquates :

```
ls -l /dev/rfcomm0
crw-rw---- 1 root dialout 216, 0 mai 28 10:38 /dev/rfcomm0
```

Seuls les utilisateurs du groupe ***dialout*** pourront utiliser **rfcomm**. Aussi, si le groupe *dialout* n'apparait pas dans les résultats de la commande *groups* pour un utilisateur, ajouter celui-ci dans le groupe *dialout* (en **root**) :

```
adduser  nom_utilisateur  dialout
```

✓ Connexion initiée par un PC :

Le PC joue le rôle de maître (*master*) et le NXT celui d'esclave (*slave*).

Initier une connexion au NXT :

```
rfcomm connect /dev/rfcomm0 00:16:53:05:80:2D
```

Un code (par défaut **1234**) vous sera demandé, pour permettre au PC de se connecter à la brique.

Quand la connexion est établie, le périphérique (*device*) ***/dev/rfcommN*** (où *N* est un entier) est créé.

L'établissement de la connexion est visible avec la commande ***hcitool*** :

```
hcitool con
Connections:
      < ACL 00:16:53:05:80:2D handle 42 state 1 lm MASTER
```

✓ Transfert de fichier en utilisant la liaison Bluetooth :

- **Avec NBC :**

```
nbc -d S=bt -b nom_ficier.rxe
```

- **Avec nexttool :**

```
nexttool /BT -download=nom_ficier.rxe
```

Remarque :

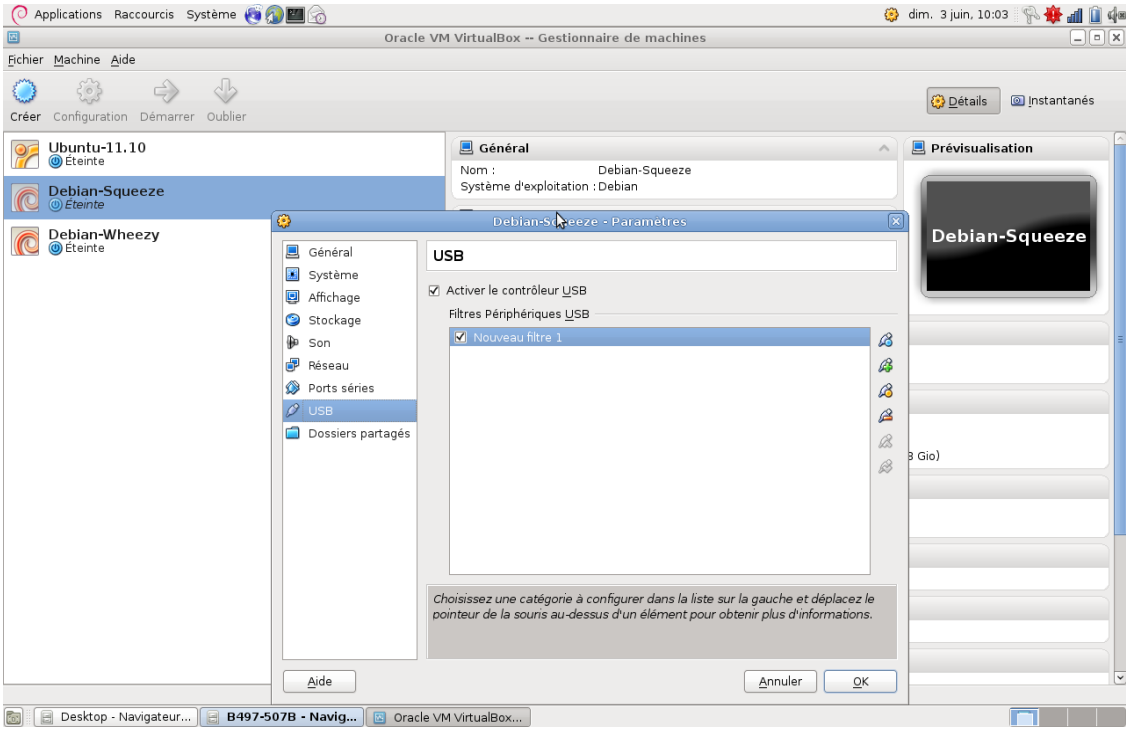
Bien que la connexion **PC → NXT** avec Bluetooth soit établie, l'envoi des fichiers vers la brique en utilisant **nbc** et **nexttoo**l a échoué !

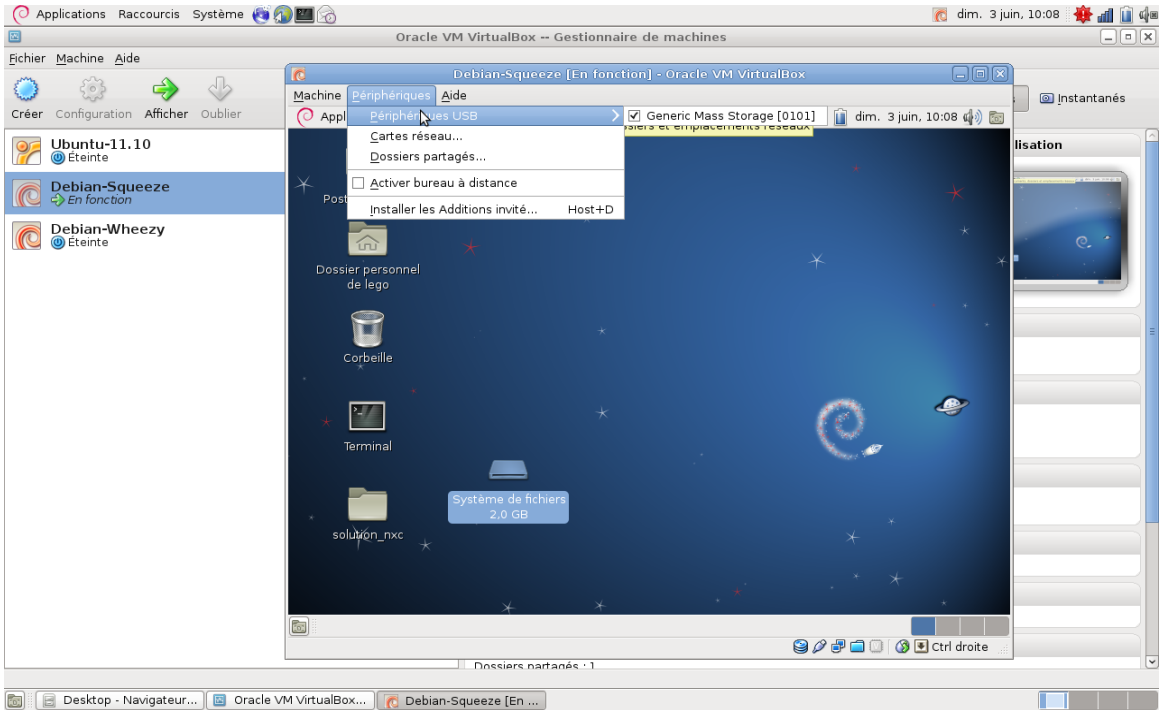
IX. Test sur des machines virtuelles avec VirtualBox (version 4.0.2):

Pour tester nos différentes solutions, on a eu recours au logiciel **VirtualBox** qui permet de créer des machines virtuelles sur lesquelles on a installé et testé nos solutions sur des environnements propres.

Il aurait fallu autoriser l'utilisation des ports **USB** aux utilisateurs des machines virtuelles installées sur ***VirtualBox*** avec la commande suivante :

```
sudo usermod -G vboxusers -a $USERNAME
```





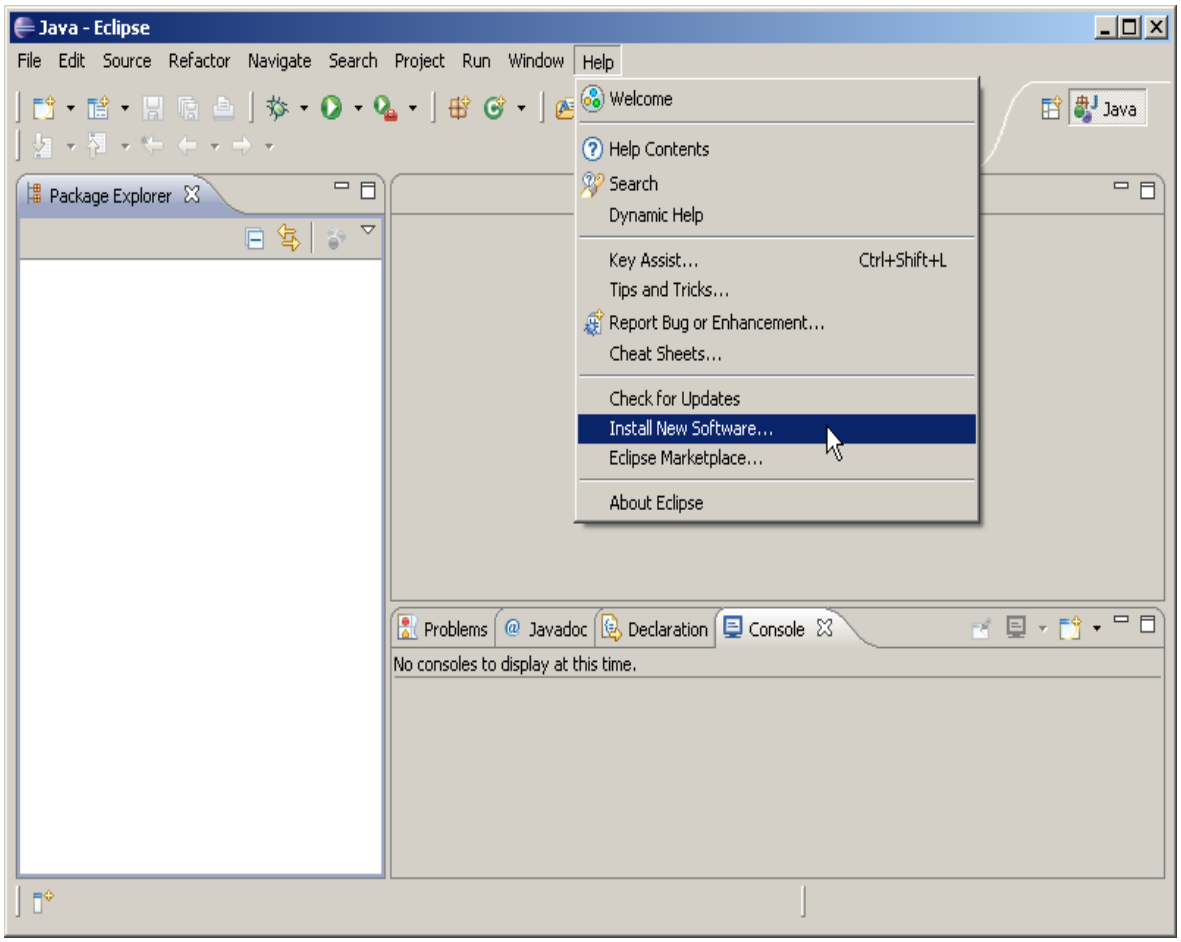
X. Solution pour la programmation en Java :

La solution choisie est **LeJOS** car elle est plus facile à mettre en place, et elle est très utilisée par la communauté Lego NXT. *(voir package et manuel d'installation **LeJOS**)*

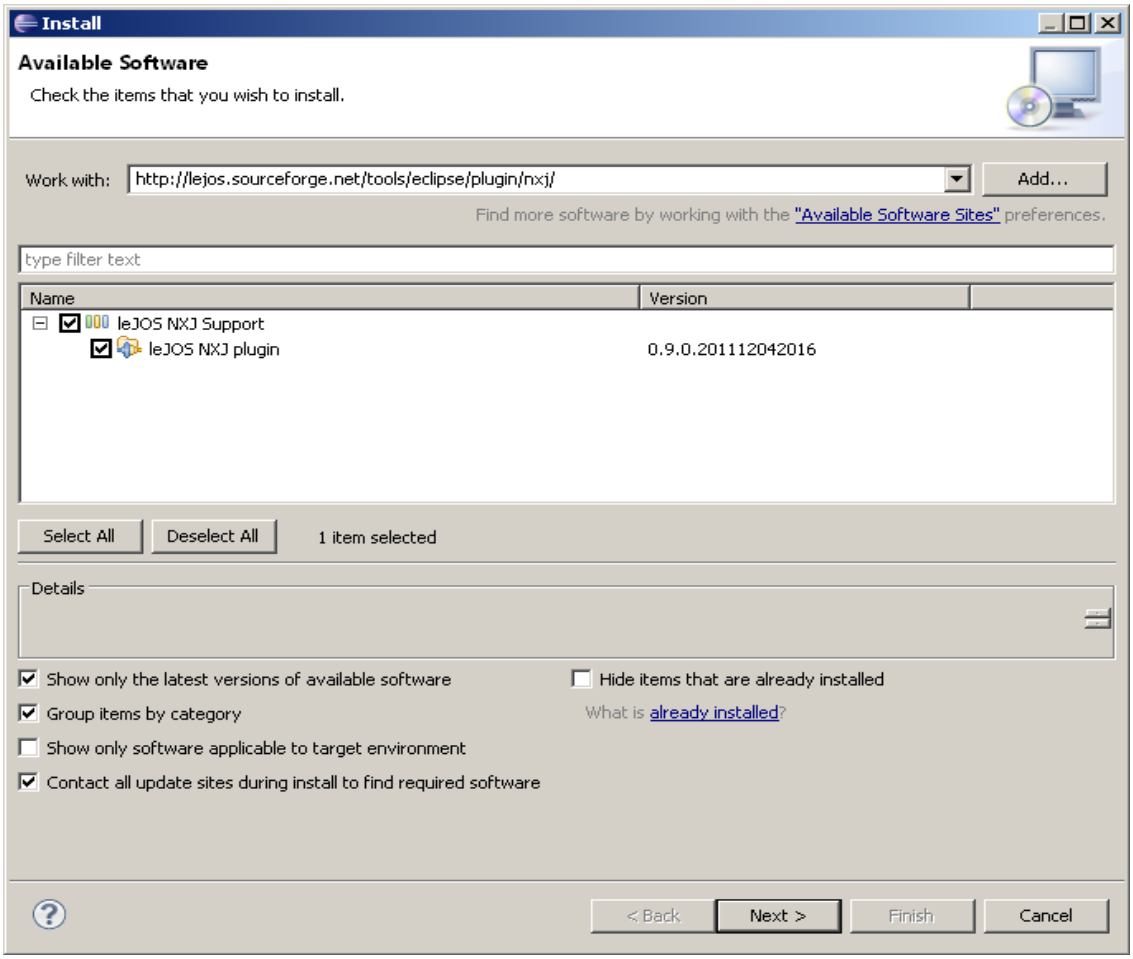
Ça nécessite d'être en mode **super-utilisateur (root)** pour éviter tout problème de droit.

❖ Installation du plugin LeJOS sous Eclipse :

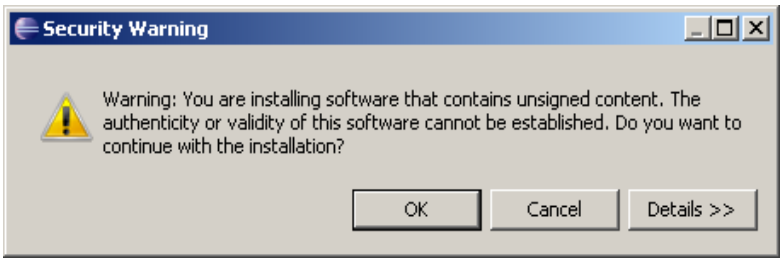
Pour installer le module LeJOS sous éclipse, allez dans le menu **«Aide» → «Installer un nouveau logiciel»**



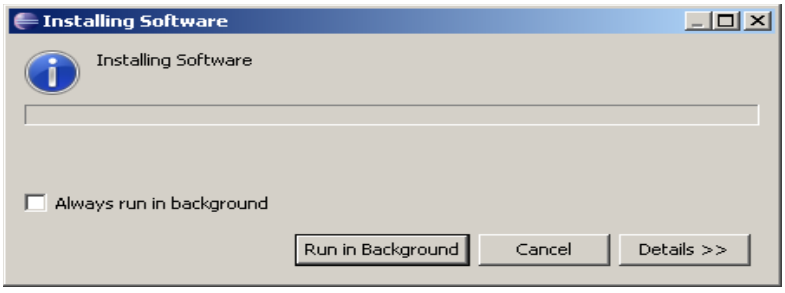
Dans la boîte de dialogue qui s'ouvre suivante tapez **"http://lejos.sourceforge.net/tools/eclipse/plugin/nxj/"** dans le champ intitulé **"travailler avec :"**. Puis appuyez sur Entrée. Eclipse va vérifier sur le site les mises à jour et affichera une liste des composants disponibles :



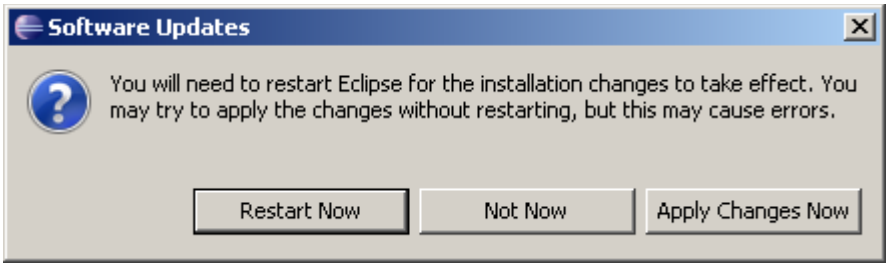
Cochez la case du plug-in lejos et cliquez sur le bouton Suivant. Confirmer à travers les prochaines pages de la boîte de dialogue d'installation et, éventuellement, cliquez sur **Terminer**. Eclipse va maintenant télécharger le plug-in.


Security Warning
Warning: You are installing software that contains unsigned content. The authenticity or validity of this software cannot be established. Do you want to continue with the installation?
OK
Cancel
Details >>

Cliquez sur « **OK** »

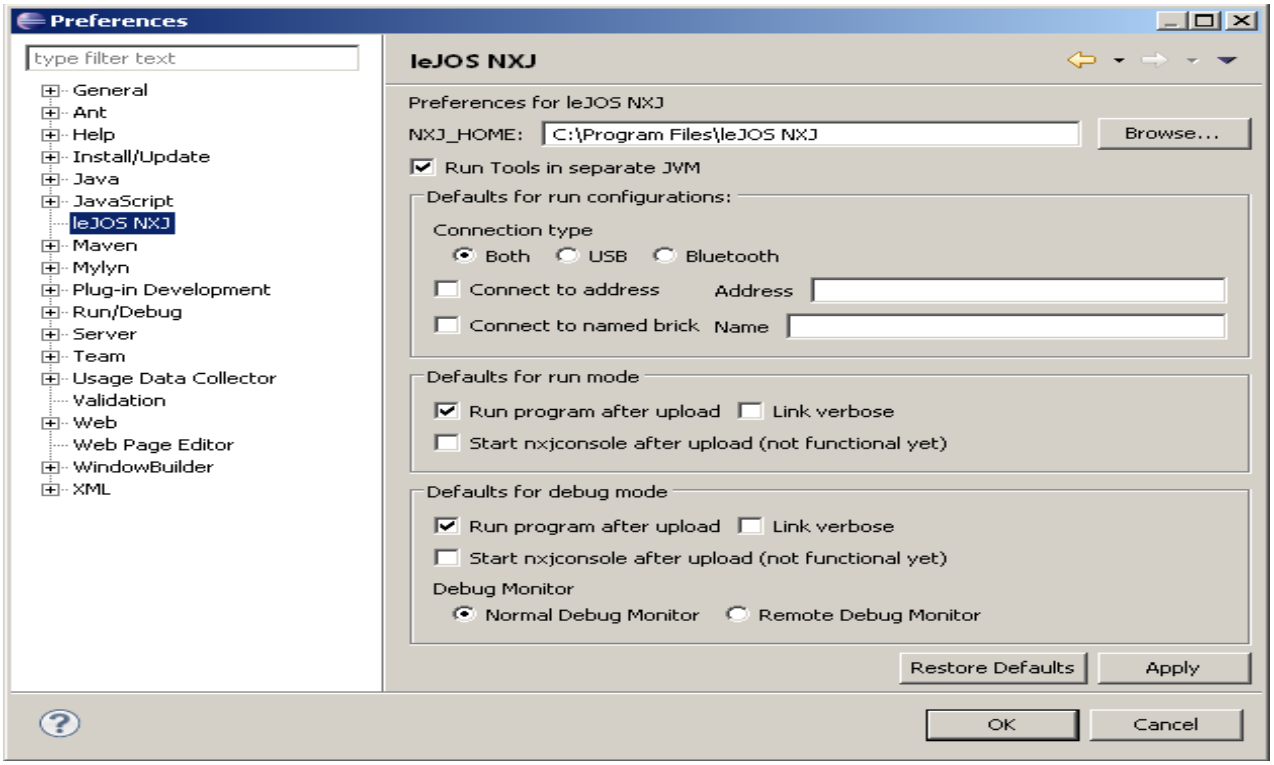


Patientez jusqu'à l'apparition de la boite de dialogue suivante :


Software Updates
You will need to restart Eclipse for the installation changes to take effect. You may try to apply the changes without restarting, but this may cause errors.
Restart Now
Not Now
Apply Changes Now

Cliquez sur « **Redémarrer maintenant** » pour redémarrer éclipse

Pour configurer le plugin pour votre système et vos préférences, cliquez sur le menu **"Fenêtre"** et sélectionnez Préférences, puis **"Lejos NXJ".**

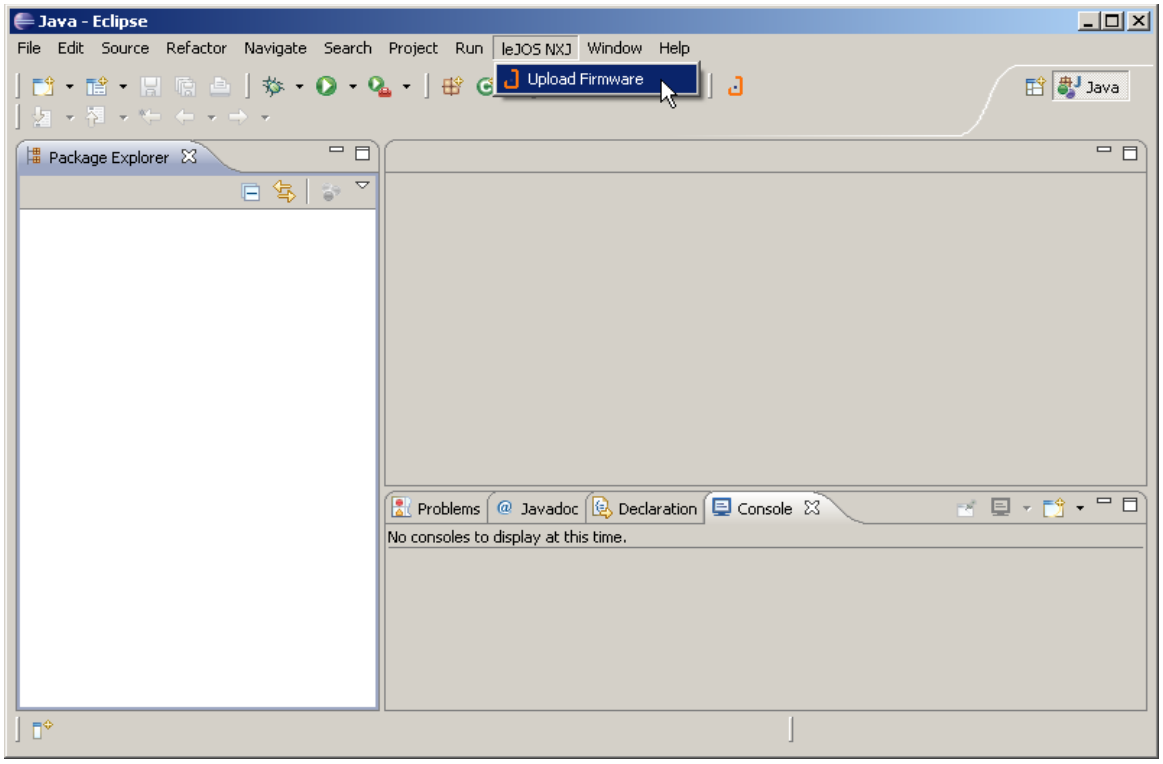


Sélectionnez le dossier dans lequel vous avez installé lejos NXJ dans **NXJ_HOME**. Sélectionnez les autres options dont vous avez besoin. Notez que ces paramètres sont les valeurs par défaut pour les configurations d'exécution créés.

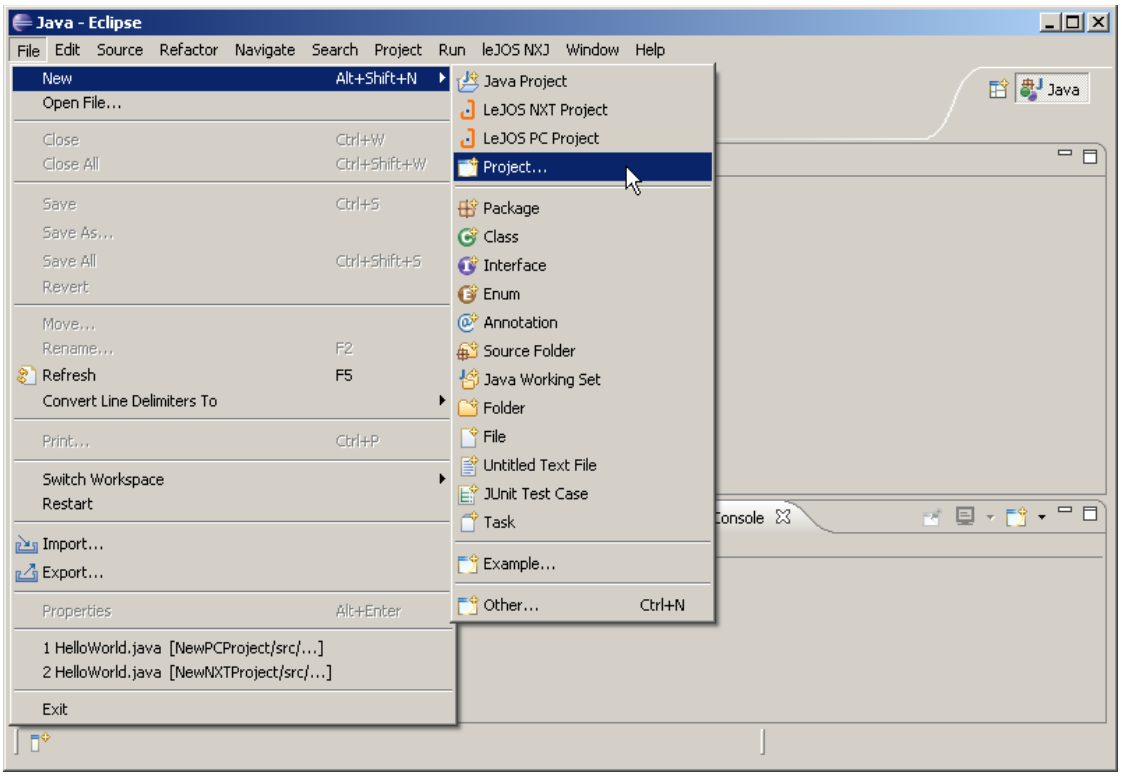
Lorsque vous avez défini vos préférences, cliquez sur **"Appliquer"** puis **"OK".**

Le plugin est maintenant installé et prêt à l'emploi.

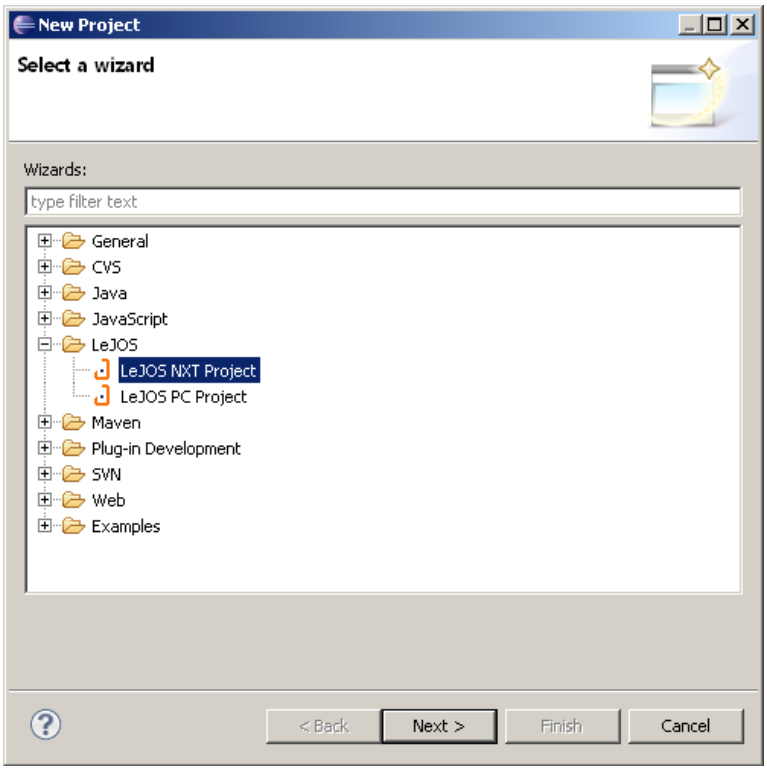
Vous pouvez télécharger le firmware lejos NXJ à votre brique NXT du plugin en cliquant sur la rubrique **"Lejos NXJ"** et en sélectionnant **"Firmware Upload"** ou en cliquant sur le bouton lejos sur la barre d'outils. Cela va démarrer l'utilitaire **NXJFlash.**


Java - Eclipse
File Edit Source Refactor Navigate Search Project Run leJOS NXJ Window Help
Upload Firmware
Java
Package Explorer
Problems Javadoc Declaration Console
No consoles to display at this time.

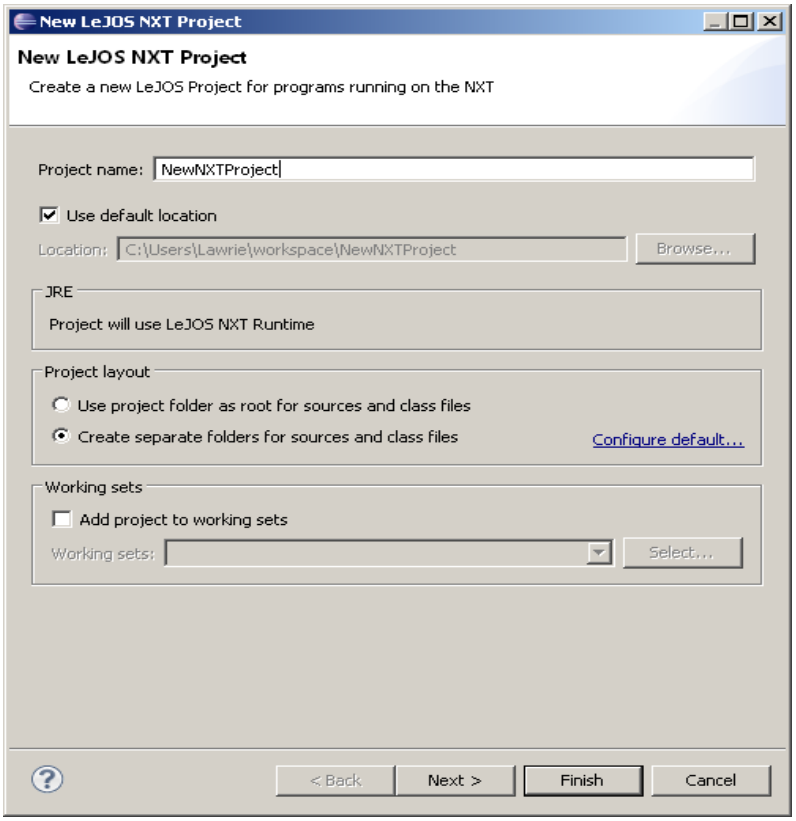
Pour créer un nouveau projet lejos NXJ en utilisant le plugin, allez dans le menu **"Fichier"** et sélectionner **"Nouveau"** puis **"Projet ..."** comme indiqué ci-dessous:



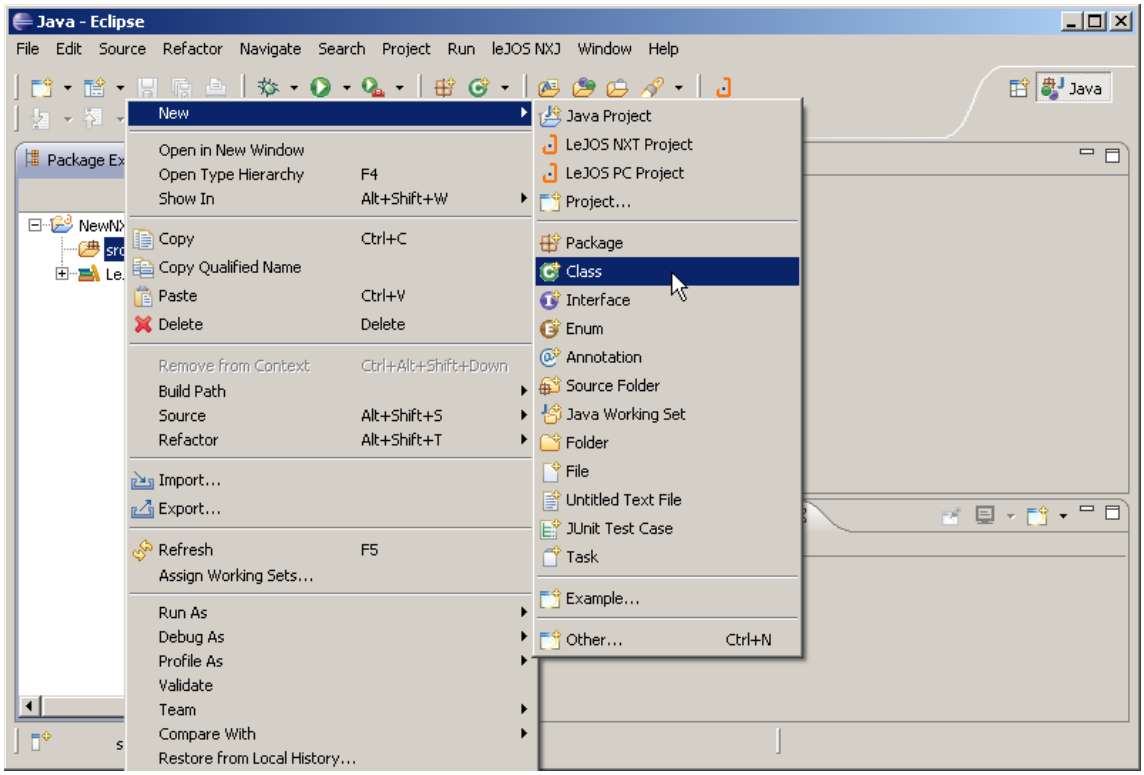
Dans la nouvelle fenêtre de dialogue, déplier la catégorie lejos et sélectionnez "Lejos NXT projet", comme indiqué ci-dessous:



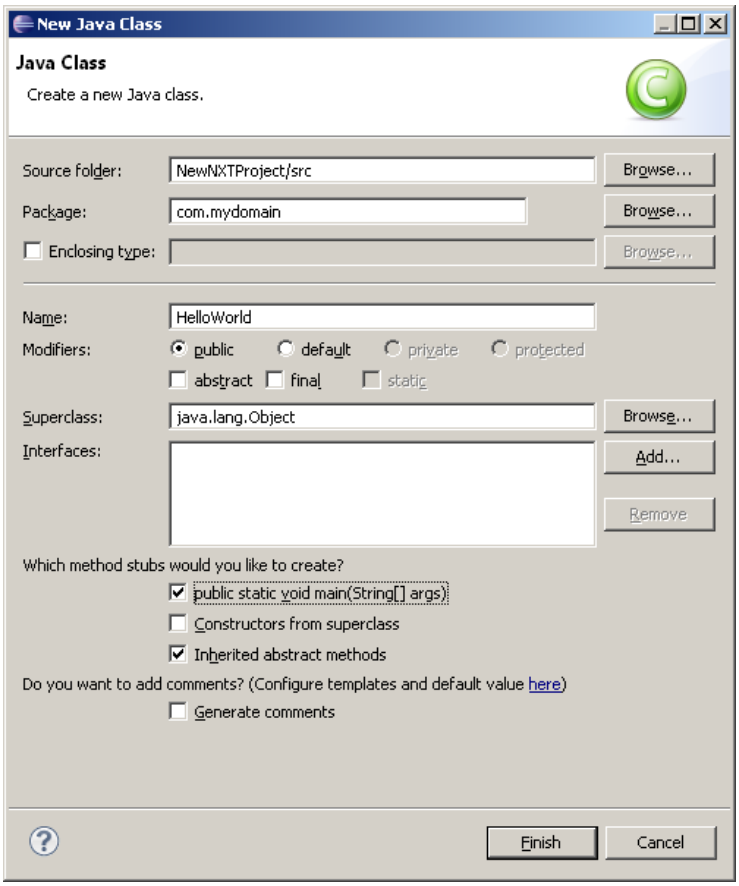
Choisissez un nom pour votre projet :



Maintenant, cliquez droit sur le dossier src et sélectionnez **"Nouveau"** puis **"classe"** dans le menu contextuel, comme illustré ci-dessous:



Cela va ouvrir l'assistant de création de nouvelles classes Java:

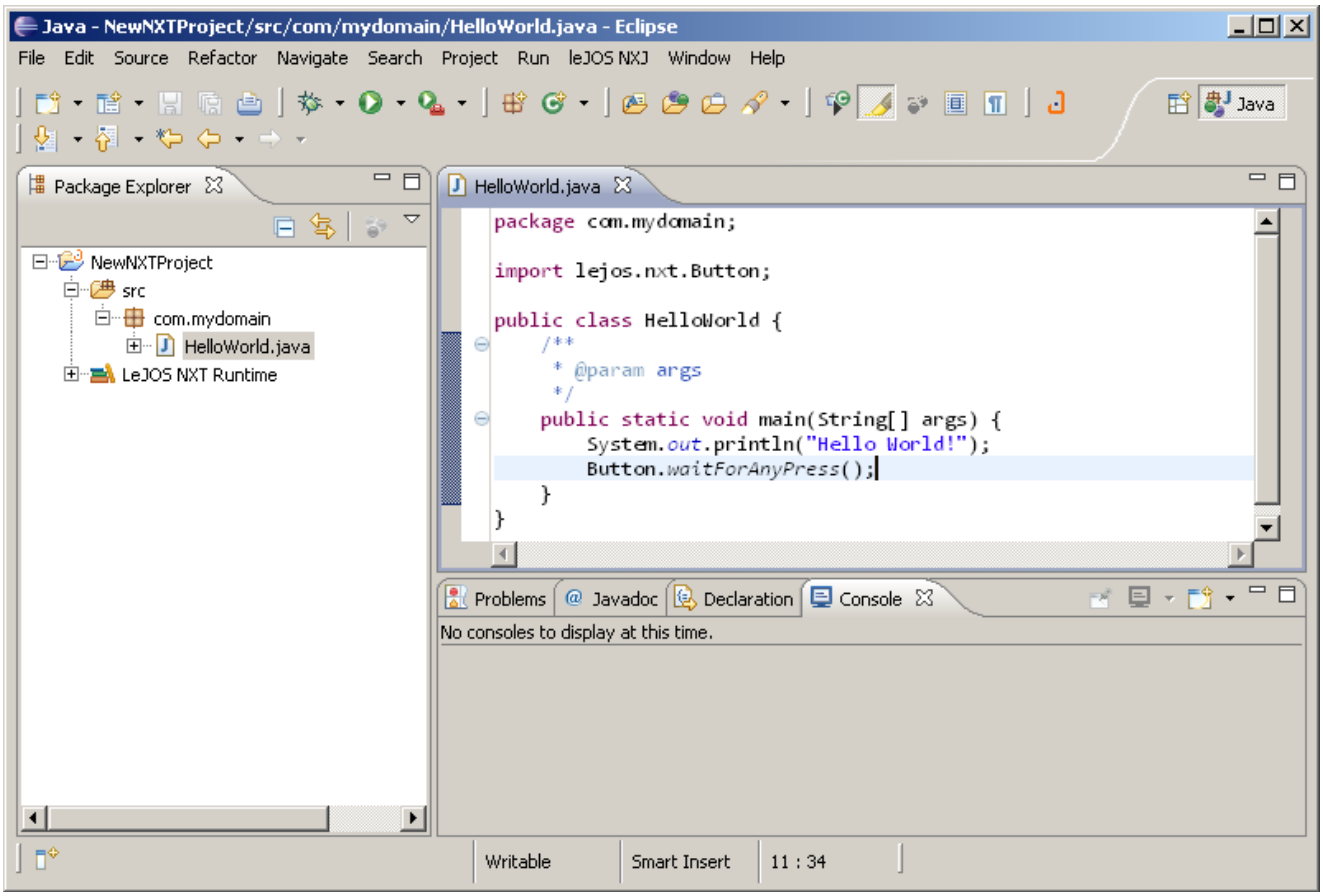


Entrez un nom de paquet et le a le nom de la classe. Dans l'exemple, le nom de la classe HelloWorld a été choisi.

Une fois que vous cliquez sur Terminer, l'éditeur de la classe s'ouvre. Maintenant, vous pouvez commencer la programmation. L'exemple de la capture d'écran montre le code suivant:

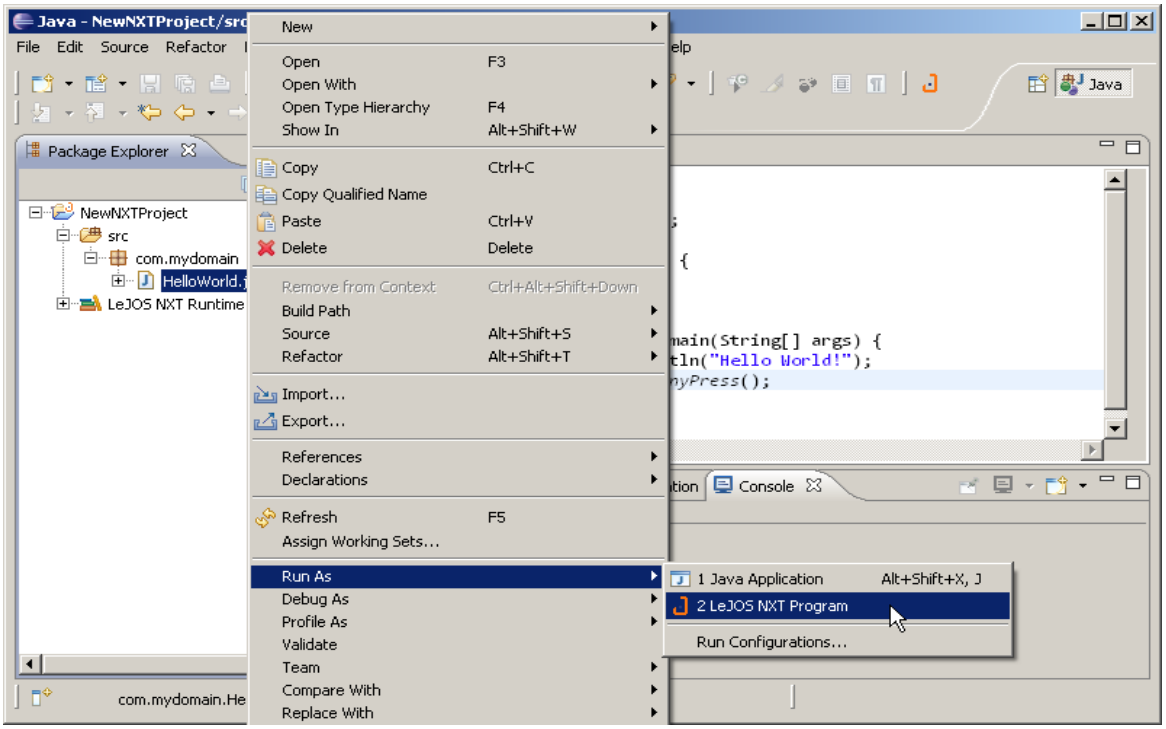
```
import lejos.nxt.Button;

public class HelloWorld {
    /**
     * @param args
     */
    public static void main(String[] args) {
        System.out.println("Hello World!");
        Button.waitForAnyPress();
    }
}
```

Le programme affichera "Hello World !" sur l'écran du NXT LCD et attendez que sur n'importe quel bouton à presser. Bien sûr, vous pouvez ajouter des classes supplémentaires et les forfaits à votre projet.

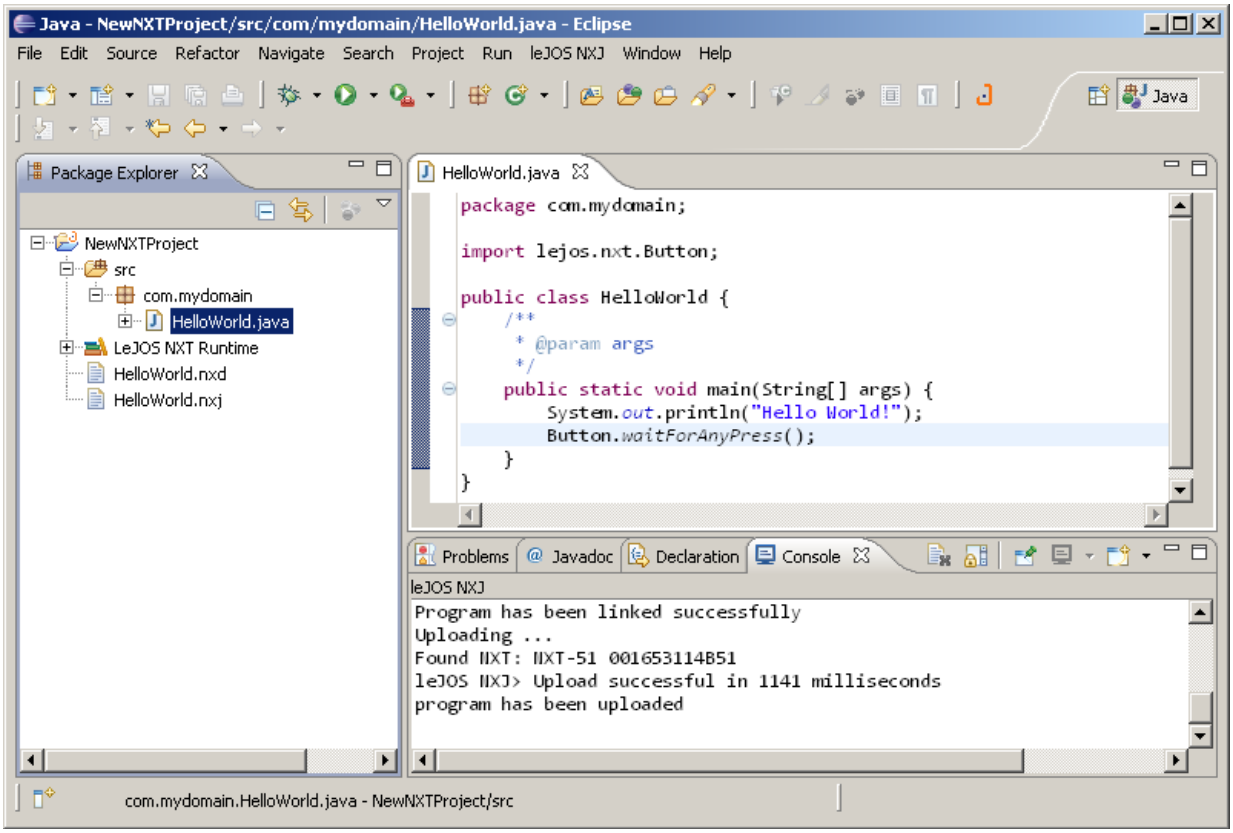


Lorsque vous êtes prêt à transférer votre programme sur la brique, cliquez-droit sur la classe qui contient la méthode principale. Sélectionnez **"Exécuter en tant que" → "Programme lejos NXT"** dans le menu contextuel.

Remarque: ne sélectionnez pas **"Exécuter en tant que"** → **"Java Application"** car cela essayez de démarrer le programme au niveau local, avec une machine virtuelle Java normale virtuelle et une exécution normale de Java. Il ne fonctionnera pas et presque certainement entraîner une erreur.



Vous verrez une sortie sur la console locale dans Eclipse. Notez la ligne après le "Hello World !" ce qui indique que le programme connecté à une brique NXT.



Vous verrez une sortie sur une console lejos NXJ. A noter également, que les deux fichiers supplémentaires ont été créés dans le répertoire racine du projet. Le fichier NXJ est le seul être de téléchargement pour la brique NXT. Le NXD est des fins de débogage.

Si vous avez sélectionné ***«Exécuter après le téléchargement»*** option, le programme démarrera sur votre NXT quand il aura fini le transfert.

XI. Conclusion :

Ce projet était une expérience très intéressante et enrichissante. Un véritable challenge et défi technique, durant lequel on a appris à surmonter les difficultés et garder notre motivation d'aller jusqu'au bout malgré quelques moments de tensions et de doute.

On a très vite compris que le sens de l'organisation, la rigueur dans le travail, notamment dans le test des solutions et la rédaction de la documentation sont des éléments essentiels pour une méthodologie efficace.

La cohésion de l'équipe, la contribution de tous ses membres et la confiance réciproque étaient des facteurs importants pour le bon déroulement et la réussite du projet.

C'était un projet qui a demandé beaucoup de recherches documentaires, et de nombreux tests sur des environnements différents pour arriver aux résultats souhaités.

XII. Annexes :

1) Vocabulaire :

Bluetooth : Bluetooth est une technologie de réseau personnel sans fils (noté WPAN pour Wireless Personal Area Network), c'est-à-dire une technologie de réseaux sans fils d'une faible portée permettant de relier des appareils entre eux sans liaison filaire.

IDE (Integrated Development Environment) ou Environnement de développement Intégré): Programme permettant de regrouper un ensemble d'outil pour le développement de logiciels

Exemple : Eclipse qui regroupe un compilateur, un éditeur de texte,...

Langage Interprété : Langage qui permet d'écrire des programmes ayant pour but d'analyser, de traduire et d'exécuter celui-ci.

Librairies : En informatique, une bibliothèque ou bibliothèque de programmes est un ensemble de fonctions utilitaires, regroupées et mises à disposition afin de pouvoir être utilisées sans avoir à les réécrire.

Licence : Une licence de logiciel est un par lequel le titulaire des droits d'auteur sur un programme définit avec son cocontractant (exploitant ou utilisateur) les conditions dans lesquelles ce programme peut être utilisé, diffusé ou modifié.

NXC : Not Exactly C

Open source : La désignation open source s'applique aux logiciels dont la licence respecte des critères précisément établis par l'Open Source Initiative, c'est-à-dire la possibilité de libre redistribution, d'accès au code source et aux travaux dérivés.

Package : Collection de programmes

Plate-forme : Une plate-forme est en informatique une base de travail à partir de laquelle on peut écrire, lire, utiliser, développer un ensemble de logiciels.

Ex : système d'exploitation

Script : Programme en langage interprété

2) Liens et sources :

http://doc.ubuntu-fr.org/pyusb

http://pyusb.sourceforge.net/docs/1.0/tutorial.html

http://wiki.zenerves.net/index.php/Nxt::usb

http://bricxcc.sourceforge.net/nbc/doc/nxtlinux.txt

http://bricxcc.sourceforge.net/nbc/

http://wiki.zenerves.net/index.php/NexTTool_manual

http://www.matth.free.fr/public/infos/legonxtdoc.php

http://www.mindstorms.rwth-aachen.de/trac/browser/tags/version-4.03/RWTHMindstormsNXT/tools/LinuxConnection

http://nxt.ivorycity.com/index.php?/archives/3-How-to-get-started-with-Linux-Bluetooth-and-the-NXT.html

http://doc.ubuntu-fr.org/virtualbox

http://lejos.sourceforge.net/

http://sourceforge.net/projects/lejos/files/lejos-NXJ

http://lejos.sourceforge.net/nxt/nxj/tutorial/Preliminaries/FirstProgram.htm

http://lejos.sourceforge.net/nxt/nxj/tutorial/Preliminaries/UsingEclipse.htm

http://lejos.sourceforge.net/rcx/api/index.html

http://nxtfr.tuxfamily.org/index.php?title=Lejos

http://www.ramytemim.com